UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/774,038	02/06/2004	Roger D. Arnold	J0658.0008	7746

38881 7590 12/21/2007

DICKSTEIN SHAPIRO LLP
1177 AVENUE OF THE AMERICAS 6TH AVENUE
NEW YORK, NY 10036-2714

EXAMINER	
POLLOCK, GREGORY A	

ART UNIT	PAPER NUMBER
4182	

MAIL DATE	DELIVERY MODE
12/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/774,038	ARNOLD ET AL.	
	Examiner	Art Unit	
	Gregory Pollock	4182	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *06 February 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the claims filed 02/06/2004.

2. Claims 1-28 have been examined.

Specification

3. The specification is objected to because of the following minor informalities:

   a. Paragraph 8, line 1, incorrect reference to "Fig. 1 (a)" should be correct to "Fig. 1".

   b. Paragraph 22, line 11 "priority thread register 310" should be corrected to "priority thread register 350".

   Appropriate correction is required.

Drawings

4. The drawings are objected to under 37 CFR 1.83(b) because they are incomplete. 37 CFR 1.83(b) reads as follows:

> When the invention consists of an improvement on an old machine the drawing must when possible exhibit, in one or more views, the improved portion itself, disconnected from the old structure, and also in another view, so much only of the old structure as will suffice to show the connection of the invention therewith.

Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as

"amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance. In order to meet compliance with the 112 paragraph 1 rejection below, the drawings will need to be modified to include a distinction between the maxtime register indices and thread values. See item 8 in the "Claim Rejections - 35 USC § 112" section below.

Claim Objections

5. Claims 14 is objected to because of the following informalities: Lines 2-4 of the claim, "a priority thread register configured to receive the internal thread value and the provide the priority thread value" should be corrected to "a priority thread register configured to receive the internal thread value and then provide the priority thread value".

6. The numbering of claims is not in accordance with 37 CFR 1.126 which requires the original numbering of the claims to be preserved throughout the prosecution. When claims are canceled, the remaining claims must not be renumbered. When new claims are presented, they must be numbered consecutively beginning with the number next following the highest numbered claims previously presented (whether entered or not).

Misnumbered claims 13-27 have been renumbered 14-28.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

8. Claim 8 rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The priority thread counter (element 210 of Figure 2) keeps track of the PTV (priority thread value) and is an integer value between 1 and N-1 ([paragraph 14, lines 8-11]). According to Figure 3, this is an index used by to indicate which maxtime register is to be active. All Figures and the specification indicated that this value is passed to the execution thread selector and used to determine the EVT (execution thread value). The execution thread value is also

an integer value between 1 and N-1 ([paragraph 14, lines 9-13]). One skilled in the art would recognize that threads are not identified by integer values but by pointers that indicated either the referencing data stream or the context register location. The specification should include the use of two separate indexes, one for the maxtime indices and one for the actual threads. From the above discussion, it is unclear how Figure 2 and 3 would differ, if the output from the priority thread selector indicated the thread not an index value. In the interest of compact prosecution, the office assumes that the embodiment represented in Figure 2 must use the same components as that in Figure 3, where an internal thread counter is used to store the index used the maxtime register. Further, it is assumed that the PTV and ETV values which are output from the priority thread selector and execution thread selector are the addresses to threads corresponding to the PTV and ETV indices.

9. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. Claim 8 is rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: the comparator and priority thread counter. The comparator is not able to increment the priority thread counter without being in some way coupled.

11. Claims 10 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claims 10 recites the limitation "wherein plurality of maxtime registers provides a maxtime value corresponding to the internal thread value. ". It is unclear if this is a new plurality of maxtime registers or the plurality of maxtime registers referenced in Claim 7. For the purpose of compact prosecution, the office assumes that these are the plurality of maxtime register reference in Claim 7. Claim 10 should be corrected to "wherein the plurality of maxtime registers provides a maxtime value corresponding to the internal thread value." to clarify the intent of the claim.

12. Claim 12 is rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: the comparator and internal thread counter. The comparator is not able to increment the internal thread counter without being in some way coupled.

13. Claims 13 recites the structural element "the priority thread counter " in line 2. There is insufficient antecedent basis for this structural element in these claims. For the purpose of compact prosecution, the office interprets this to mean "the internal thread counter". Appropriate correction is required.

14. Claim 14 rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential steps, such omission amounting to a gap between the steps. See MPEP § 2172.01. The omitted steps are: communication from the comparator to the priority thread register that the maxtime value in not equal to zero.

15. Claim 14 rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: the comparator and the priority thread register. In order for the comparator to communicate the state of the maxtime value, it must be coupled to the priority thread register.

16. Claims 16 recites the structural elements “the block values” in line 3 and “the comparison result” in lines 3-4. There is insufficient antecedent basis for these structural elements in these claims. For the purpose of compact prosecution, the office assumes that Claim 16 is dependent on Claim 15. Appropriate action should be taken to reflect the intent of the claim.

17. Claim 16 is rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential elements, such omission amounting to a gap between the elements. See MPEP § 2172.01. The omitted elements are: thread block checker, comparator, and execution thread register. These elements are required to generate the block values, comparison result, and execution thread

value received by the controller. For the purpose of compact prosecution, the office assumes that Claim 16 is dependent on Claim 15. Appropriate action should be taken to reflect the intent of the claim.

18. Claim 16 rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: the controller and the thread block checker, comparator, and execution thread register. These all must be coupled to the controller for the block values, comparison result, and execution thread value to be received. Further, Figure 4 shows the execution thread register receiving the next execution thread value from the controller. This is not mentioned within Claim 16. For the purpose of compact prosecution, the office assumes that Claim 16 is dependent on Claim 15. Appropriate action should be taken to reflect the intent of the claim.

Claim Rejections - 35 USC § 102

19. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

20. Claims 1, 2, 3, 19, 20, 24, and 25 are rejected under 35 U.S.C. 102 (b) as being anticipated by Rudd ET al. (U.S. Application No. 10/173334, Date Filed: June 14, 2002).

As per claim 1, Rudd et al. teaches a thread selection unit (thread control logic, element 270, Figure 2, whose details are shown in Figure 3) for a multithreaded processor ([Abstract, line 2]) having a plurality of active threads ([paragraph 16, lines 2-3], the thread selection unit comprising:

a priority thread selector (thread prioritizer (element 310) and switch enabler (element 320), Figure 3) configured to generate a priority thread value (absolute thread priority, element 354, Figure 3).

an execution thread selector (next thread selection, element 330, Figure 3) coupled to receive the priority thread value and to generate an execution thread value (next threads, element 360, Figure 3) associated with an execution thread.

As per claim 2, the rejection of claim 1 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the execution thread selector is configured to select the priority thread as the execution thread when the priority thread is unblocked. ([paragraph 29, lines 13-17], where “the current thread waiting for memory or I/O access” is an example of a blocked thread)

As per claim 3, the rejection of claim 1 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread selector selects the priority thread without regards to the actions of the execution thread selector. (Figure 3, where next threads (output from next thread selector) is not an input to either the thread prioritizer or switch enabler)

As per claim 19, Rudd et al. teaches a method of selecting an execution thread from a plurality of active threads in a multithreaded processor ([Abstract, lines 1-2]), the method comprising:

selecting a priority thread;(thread prioritizer (element 310) and switch enabler (element 320), Figure 3)

selecting the priority thread as the execution thread, when the priority thread is unblocked.([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

As per claim 20, the rejection of claim 19 has been addressed.
Rudd et al. further teaches a method further comprising selecting a non-priority thread as the execution thread when the priority thread is

blocked..([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

As per claim 24, Rudd et al. teaches a thread selection unit for selecting an execution thread from a plurality of active threads in a multithreaded processor, the thread selection unit comprising:

means for selecting a priority thread;(thread prioritizer (element 310) and switch enabler (element 320), Figure 3)

means for selecting the priority thread as the execution thread, when the priority thread is unblocked..([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

As per claim 25, the rejection of claim 24 has been addressed.
Rudd et al. further teaches a thread selection unit further comprising means for selecting a non-priority thread as the execution thread when the priority thread is blocked. ([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

Claim Rejections - 35 USC § 103

21. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

22. Claims 4-14, 21-23, and 26-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rudd ET al. (U.S. Application No. 10/173334, Date Filed: June 14, 2002).

As per claim 4, the rejection of claim 1 has been addressed.

Rudd et al. teaches a thread selection unit wherein the priority thread selector comprises a plurality of maxtime registers (threshold values (element 522), Figure 5, where one skilled in the art would understand that these values must be stored in registers for use in counter comparisons),

Rudd et al. does not teach a thread selection unit wherein each active thread has an associated maxtime register.

Rudd et al. uses forward progress counters (elements 322 and 324) in Figure 3, which are associated with each active thread ([paragraph 28, lines 1-2]). The forward progress counters count events which may include weighted processor

cycles ([paragraph 28, lines12-29]). Thresholds are used to determine when a forward progress counter has reached a limit where the priority thread may be switched. Each threshold value can be associated with a forward progress counters ([paragraph 35, lines 12-21] and [paragraph 36, lines 5-9]). Rudd et al. further teaches that it was well know in the art at the time of the invention that in sequential multi-threaded operations, "threads are switched when the currently executing thread executes for a certain period or reaches a point when it cannot proceed, such as waiting for a memory access or an I/O transfer to finish." ([paragraph 2, lines 6-15]).

One skilled in the art would interpret this to mean that each thread was allotted a certain number of cycles (maxtime counts) to be the priority thread unless it was a blocked thread. It would have been obvious, and well known, to one skilled in the art at the time of the invention to have modified Rudd et al. to have only used one forward progress counter and to have compared it to the thresholds which are associated with that counter, and hence each thread. This would have greatly simplified the design of Rudd et al. The invention of Rudd et al. was expanding the previously well know use of a counter, maxtime registers, and comparators to a more dynamic system which used multiple counters which could be triggered off of different events beyond just clock cycles The invention of Rudd et al. could have been reverted to a more simple design for the purpose of cost.

As per claim 5, the rejection of claim 4 has been addressed.

Rudd et al. further teaches a thread selection unit wherein the priority thread selector further comprises a priority thread counter (thread prioritizer (element 310) configured to provide the priority thread value to the execution thread selector (absolute thread priority, element 354, Figure 3).

As per claim 6, the rejection of claim 5 has been addressed.

Rudd et al. further teaches a thread selection unit wherein plurality of maxtime registers provides a maxtime value (threshold (element 522), Figure 5) corresponding to the priority thread value (thresholds can be associated with individual forward progress counters [paragraph 35, lines 12-21] and [paragraph 36, lines 5-9] which have corresponding threads [paragraph 28, lines 1-2]).

As per claim 7, the rejection of claim 6 has been addressed.

Rudd et al. further teaches a thread selection unit wherein the priority thread selector further comprises: a counter (forward progress counters (elements 322 and 324), Figure 3);

Rudd et al. does not specifically teach a comparator coupled to the counter and the plurality of maxtime registers, wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers.

However, one skilled in the art would recognize that a comparator must be used when Rudd et al. performs a comparison ([paragraph 36, lines 4-12]) of the forward progress counters, which are associated with each thread ([paragraph 28, lines 1-2]), to its associated threshold value ([paragraph 35, lines 12-13]).

As per claim 8, the rejection of claim 7 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread counter is incremented (a new thread can be switched in [paragraph 35 lines 7-10]) and the counter is reset (initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized) when the count value equals the maxtime value (Figure 5, when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]).

As per claim 9, the rejection of claim 4 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread selector further comprises an internal thread counter (thread prioritizer (element 310)) configured to provide an internal thread value to the maxtime registers. (the thread prioritizer has corresponding forward progress counters [paragraph 28,

lines 1-2]. Therefore, when a thread is the active thread the corresponding forward progress counter is active)

As per claim 10, the rejection of claim 9 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the plurality of maxtime registers provides a maxtime value (threshold (element 522), Figure 5) corresponding to the internal thread value (thresholds can be associated with individual forward progress counters [paragraph 35, lines 12-21] and [paragraph 36, lines 5-9] which have corresponding threads [paragraph 28, lines 1-2]).

As per claim 11, the rejection of claim 10 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread selector further comprises: a counter (forward progress counters (elements 322 and 324), Figure 3);

Rudd et al. does not specifically teach a comparator coupled to the counter and the plurality of maxtime registers, wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers.

However, one skilled in the art would recognize that a comparator must be used when Rudd et al. performs a comparison ([paragraph 36, lines 4-12]) of the

forward progress counters, which are associated with each thread ([paragraph 28, lines 1-2]), to its associated threshold value ([paragraph 35, lines 12-13]).

As per claim 12, the rejection of claim 11 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the internal thread counter is incremented (a new thread can be switched in [paragraph 35 lines 7-10]) and the counter is reset (initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized) when the count value equals the maxtime value. (Figure 5, where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22])

As per claim 13, the rejection of claim 11 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread counter is incremented (a new thread can be switched in [paragraph 35 lines 7-10]) and the counter is reset (initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized) when the

maxtime value equals zero. (Figure 5, where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 throught paragraph 36 line 22]. If the forward progress counter is zero, the thread can be switched immediately.)

As per claim 14, the rejection of claim 12 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the priority thread selector further comprises a priority thread register (thread prioritizer (element 310), which contains a register of prioritized threads) configured to receive the internal thread value (the thread prioritizer performs the functions of both the internal thread counter and priority thread register) and then provide the priority thread value (absolute thread priority, element 354, Figure 3), wherein the priority thread registers stores the internal thread value when the maxtime value is not equal to zero. (Figure 5, where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]. It the forward progress counter is zero, the thread can be switched immediately.)

As per claim 21, the rejection of claim 20 has been addressed.
Rudd et al. further teaches selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles. (a new thread can be switched in [paragraph 35 lines 7-10] when the forward

progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]).

Rudd et al. does not teach a method wherein the selecting a priority thread comprises: assigning a maxtime value for each active thread;

Rudd et al. does teach (threshold values (element 522), Figure 5. Rudd et al. uses forward progress counters (elements 322 and 324) in Figure 3, which are associated with each active thread ([paragraph 28, lines 1-2]). The forward progress counters count events which may include weighted processor cycles ([paragraph 28, lines 12-29]). Thresholds are used to determine when a forward progress counter has reached a limit where the priority thread may be switched. Each threshold value can be associated with a forward progress counters ([paragraph 35, lines 12-21] and [paragraph 36, lines 5-9]). Rudd et al. further teaches that it was well know in the art at the time of the invention that in sequential multi-threaded operations, “threads are switched when the currently executing thread executes for a certain period or reaches a point when it cannot proceed, such as waiting for a memory access or an I/O transfer to finish.” ([paragraph 2, lines 6-15]).

One skilled in the art would interpret this to mean that each thread was allotted a certain number of cycles (maxtime counts) to be the priority thread unless it was

a blocked thread. It would have been obvious, and well known, to one skilled in the art at the time of the invention to have modified Rudd et al. to have only used one forward progress counter and to have compared it to the thresholds which are associated with that counter, and hence each thread. This would have greatly simplified the design of Rudd et al. The invention of Rudd et al. was expanding the previously well know use of a counter, maxtime registers, and comparators to a more dynamic system which used multiple counters which could be triggered off of different events beyond just clock cycles. The invention of Rudd et al. could have been reverted to a more simple design for the purpose of cost.

As per claim 22, the rejection of claim 21 has been addressed.
Rudd et al. further teaches a method wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: of incrementing a priority thread counter (a new thread can be switched in [paragraph 35 lines 7-10]) when a count value equals the maxtime value (Figure 5, when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]) corresponding to the priority thread;

and resetting a counter when the count value equals the maxtime value (initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress

counters are used, they do not need to be reset until they are to be used. At which time they are initialized) corresponding to the priority thread.

As per claim 23, the rejection of claim 21 has been addressed.
Rudd et al. further teaches a method wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: incrementing an internal thread counter (thread prioritizer (element 310)) when a count value equals the maxtime value corresponding to a internal thread value (Figure 5, when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]).

resetting a counter when the count value equals the maxtime value corresponding to the priority thread;(initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized)

and setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero.(Figure 5, where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through

paragraph 36 line 22]. If the forward progress counter is zero, the thread can be switched immediately.)

As per claim 26, the rejection of claim 25 has been addressed.
Rudd et al. further teaches a means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles. (a new thread can be switched in [paragraph 35 lines 7-10] when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]).

Rudd et al. does not teach a thread selection unit wherein the means for selecting a priority thread comprises: means for assigning a maxtime value for each active thread; (threshold values (element 522), Figure 5. Rudd et al. uses forward progress counters (elements 322 and 324) in Figure 3, which are associated with each active thread ([paragraph 28, lines 1-2]). The forward progress counters count events which may include weighted processor cycles ([paragraph 28, lines 12-29]). Thresholds are used to determine when a forward progress counter has reached a limit where the priority thread may be switched. Each threshold value can be associated with a forward progress counters ([paragraph 35, lines 12-21] and [paragraph 36, lines 5-9]). Rudd et al. further teaches that it was well know in the art at the time of the invention that in sequential multi-threaded operations, “threads are switched when the currently

executing thread executes for a certain period or reaches a point when it cannot proceed, such as waiting for a memory access or an I/O transfer to finish." ([paragraph 2, lines 6-15]).

One skilled in the art would interpret this to mean that each thread was allotted a certain number of cycles (maxtime counts) to be the priority thread unless it was a blocked thread. It would have been obvious, and well known, to one skilled in the art at the time of the invention to have modified Rudd et al. to have only used one forward progress counter and to have compared it to the thresholds which are associated with that counter, and hence each thread. This would have greatly simplified the design of Rudd et al. The invention of Rudd et al. was expanding the previously well know use of a counter, maxtime registers, and comparators to a more dynamic system which used multiple counters which could be triggered off of different events beyond just clock cycles The invention of Rudd et al. could have been reverted to a more simple design for the purpose of cost.

As per claim 27, the rejection of claim 26 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing a priority thread counter (a new thread can be switched in [paragraph 35 lines 7-10]) when a count value equals the maxtime value

corresponding to the priority thread;(Figure 5, when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22])

and means for resetting a counter when the count value equals the maxtime value corresponding to the priority thread.(initialized to zero, Figure 8, element 810 and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized)

As per claim 28, the rejection of claim 26 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing an internal thread counter (a new thread can be switched in [paragraph 35 lines 7-10]) when a count value equals the maxtime value corresponding to a internal thread value; (the thread prioritizer has corresponding forward progress counters [paragraph 28, lines 1-2]. Therefore, when a thread is the active thread the corresponding forward progress counter is active)

means for resetting a counter when the count value equals the maxtime value corresponding to the priority thread; (initialized to zero, Figure 8, element 810

and [paragraph 43, line 1-4], where one skilled in the art would recognize that because multiple forward progress counters are used, they do not need to be reset until they are to be used. At which time they are initialized)

and means for setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero.; (Figure 5, where a new thread can be switched in when the forward progress counter reaches the threshold value [paragaphs 35 lines 10 through paragraph 36 line 22]. If the forward progress counter is zero, the thread can be switched immediately.)

23. Claims 15 -18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rudd et al. (U.S. Application No. 10/173334, Date Filed: June 14, 2002) in view of Borkenhagen et al. (U.S. Patent No. 6,567,839, Date Issued: May 20, 2003).

As per claim 15, the rejection of claim 1 has been addressed.
Rudd et al. teaches a thread selection unit wherein the execution thread value (next thread (element 360)) is generated from the execution thread selector. (next thread selection, element 330, Figure 3)

Rudd et al. does not teaches the specific details of the next thread selector ((element 330) Figure 3).

Borkenhagen et al. teaches a thread selection unit wherein the execution thread selector (thread switch logic hardware (element 400) or Figure 4A and B) comprises: a thread block checker (thread switch control register (element 410)) configured to provide a plurality of block values ([column 12, line 64 through column 13 line 3]), wherein each active thread has a corresponding block value (column 13 lines 2-3]); an execution thread register (storage control unit (element 200) of Figure 4B, where the execution thread register are part of the instruction pipeline for execution of instructions) configured to provide the execution thread value (input "A" in Figures 4A and output "A" in Figure 4B));

It would have been obvious to one skilled in the art at the time of the invention to have used the thread switch unit implemented in Borkenhagen et al. as the next thread selection unit of Rudd et al. Further, it would have been obvious to one skilled in that art that ("The next thread selector "(element 330) Figure 3) could be combined with the thread switch logic of Borkenhagen et al.(Figures 4A and 4B) to provide a simple comparison using a comparator, which would be know to someone skilled in the art, of the priority thread value (Rudd et al., absolute thread value (element 354) of Figure 3)) and the execution thread value (Borkenhagen et al. input "A" in Figures 4A and output "A" in Figure 4B). The purpose of the comparison would be to more quickly control determination of the

next thread to be selected by utilizing the immediate result of the comparison in the logic which the next thread selector outputs.

As per claim 16, the rejection of claim 15 has been addressed.
Rudd et al. does not teach a thread selection unit to receive the priority thread values (absolute thread (element 354, Figure 3)) that generates a next execution thread value for the execution thread register. (next threads (element 360, Figure 3)

Rudd et al. does not teach the thread selection unit wherein the execution thread selector further comprises a controller coupled to receive the block values ([column 12, line 64 through column 13 line 3]), the comparison result, and the execution thread value.

Borkenhagen et al teaches the thread selection unit wherein the execution thread selector further comprises a controller (Borkenhagen et al., thread switch controller (element 450) of Figure 4A) coupled to receive the block values (Borkenhagen et al., thread switch control register is coupled to the thread switch controller in Figure 4A) coupled to receive the block values, the priority thread value, the comparison result, and the execution thread value and configured to generate a next execution thread value for the execution thread register. (thread

switch controller (element 450) of Figure 4A is shown receiving all inputs required for determining the active thread.)

It would have been obvious to one skilled in the art at the time of the invention to have used the thread switch controller in Borkenhagen et al. as the next thread selection unit of Rudd et al. The purpose would have been able to use a controller to make logical decisions based the all available thread status (block information, the priority thread value, the current thread being executed, and all active threads). The purpose of using a controller (as opposed to hardware logic) is that the logic can be changed without a hardware design. This would reduce cost and lead time when implementing desired logic changes in future enhancements. Note that next thread selector (element 330) of Figure 3 in Rudd et al. has all information available to it in the execution information (element 352) and configuration (element 350) for implementing the logic of Borkenhagen et al.

As per claim 17, the rejection of claim 16 has been addressed.
Rudd et al. further teaches a thread selection unit wherein the controller generates the next execution thread value to be equal to the priority thread value when the priority thread is not blocked. ([paragraph 29, lines 13-17] and [paragraph 32], where “the current thread waiting for memory or I/O access” is an example of a blocked thread)

As per claim 18, the rejection of claim 17 has been addressed.

Rudd et al. further teaches a thread selection unit wherein the controller generates the next execution thread value to not be equal to the priority thread value when the priority thread is blocked.([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

As per claim 19, Rudd et al. teaches a method of selecting an execution thread from a plurality of active threads in a multithreaded processor ([Abstract, lines 1-2]), the method comprising:

selecting a priority thread;(thread prioritizer (element 310) and switch enabler (element 320), Figure 3)

selecting the priority thread as the execution thread, when the priority thread is unblocked.([paragraph 29, lines 13-17] and [paragraph 32], where "the current thread waiting for memory or I/O access" is an example of a blocked thread)

Claim Rejections - 35 USC § 103

24. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

25. Claims 1-14 and 19-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vaitzblit et al. (U.S. Patent No. 5,528,513, Date Published: June 18, 1996) in view of Sager (U.S. Application No. 10/361,368, Date Published: August 14, 2003).

As per claim 1, Vaitzblit et al. teaches a thread selection unit (general purpose ready queue" (element 108) Figure 2,) for a multithreaded processor ([column 3, lines 38-43]) having a plurality of active threads ([column 3, lines 38-43]), the thread selection unit comprising:

an execution thread selector (scheduler (element 53) in Figure 1) coupled to receive the priority thread value (supports a general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100) and to generate an execution thread value (real-time streams [column 3, lines 17-26]) associated with an execution thread.

Vaitzblit et al. does not teach a priority thread selector configured to generate a priority thread value.

Sager teaches a priority thread selector (apparatus for controlling the processing priority [¶61 lines 2-3] and see Figure 13 which can be used to implement the flow diagram of Figure 9) configured to generate a priority thread value. (thread priority signal [¶61 lines 9-12])

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 2, the rejection of claim 1 has been addressed.

Vaitzblit et al. further teaches a thread selection unit wherein the execution thread selector is configured to select the priority thread as the execution thread when the priority thread is unblocked. ([column 5, lines 22-23])

As per claim 3, the rejection of claim 1 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector selects the priority thread without regards to the actions of the execution thread selector.

Sager teaches a thread selection unit wherein the priority thread selector selects the priority thread without regards to the actions of the execution thread selector. (the flow diagram of Figure 9 has only threads as inputs. Further, Figure 13 does not have the execution thread selector as an input.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) select the priority thread without regards to the actions of the execution thread selector, and output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue

(element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 4, the rejection of claim 1 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector comprises a plurality of maxtime registers, wherein each active thread has an associated maxtime register.

Sager teaches a thread selection unit wherein the priority thread selector comprises a plurality of maxtime registers ("initialize priority period" as found in element 909 in Figure 9), wherein each active thread has an associated maxtime register. ("for each thread" as found in element 909 in Figure 9)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the

general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 5, the rejection of claim 4 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector further comprises a priority thread counter.

Sager teaches a thread selection unit wherein the priority thread selector further comprises a priority thread counter ("assign priority to the other thread" logic (block 913) would use a counter similar to that shown in Figure 14 as elements 1403 and 1405.) configured to provide the priority thread value to the execution thread selector (the "thread priority control" block (element 1301) is shown providing its output to "allocate" (element 1311) and "schedule/displatch" (element 1321) units).

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 6, the rejection of claim 5 has been addressed.
Vaitzblit et al. in does not teach a thread selection unit wherein plurality of maxtime registers provides a maxtime value corresponding to the priority thread value.

Sager teaches a thread selection unit wherein plurality of maxtime registers provides a maxtime value ("priority period" as found in element 909 in Figure 9), corresponding to the priority thread value. ("for each thread" as found in element 909 in Figure 9)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 7, the rejection of claim 6 has been addressed.

Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector further comprises: a counter; and a comparator coupled to the counter and the plurality of maxtime registers, wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers.

Sager teaches a thread selection unit wherein the priority thread selector further comprises: a counter; ("current priority period expired" logic (block 913) would use a counter similar to that shown in Figure 14 as elements 1403 and 1405.) a comparator ("current priority period expired" logic (block 913) would use a comparator similar to that shown in Figure 14 as elements 1415.) coupled to the counter and the plurality of maxtime registers ("current priority period expired" decision block would require the initialized priority periods values of block 905 and counter be coupled to the comparator in order to carry out the logic), wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers. ("current priority period expired" logic (element 913))

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the

general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 8, the rejection of claim 7 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread counter is incremented and the counter is reset when the count value equals the maxtime value.

Sager teaches a thread selection unit wherein the priority thread counter is incremented ("assign priority to the other thread" (element 917)) and the counter is reset (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.) when the counter is reset when the count value equals

the maxtime value ("the priority period for a thread has expired" [paragraph 51, lines 19-22]).

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 9, the rejection of claim 4 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector further comprises an internal thread counter configured to provide an internal thread value to the maxtime registers.

Sager teaches a thread selection unit wherein the priority thread selector further comprises an internal thread counter ("assign priority to the other thread" logic (block 913) would use a counter similar to that shown in Figure 14 as elements 1403 and 1405.) configured to provide an internal thread value to the maxtime registers. (the "determine next priority period for each thread" block (element 921)).

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 10, the rejection of claim 9 has been addressed.
Vaitzblit et al. further teaches a thread selection unit wherein plurality of maxtime registers provides a maxtime value corresponding to the internal thread value.

Sager teaches a thread selection unit wherein the plurality of maxtime registers provides a maxtime value ("priority period" as found in element 909 in Figure 9) corresponding to the internal thread value ("for each thread" as found in element 909 in Figure 9)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously

change their state in response to the changes in the other threads without doing any useful work.

As per claim 11, the rejection of claim 10 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the priority thread selector further comprises: a counter; and a comparator coupled to the counter and the plurality of maxtime registers, wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers.

Sager teaches a thread selection unit wherein the priority thread selector further comprises: a counter ("current priority period expired" logic (block 913) would use a counter similar to that shown in Figure 14 as elements 1403 and 1405.); a comparator ("current priority period expired" logic (block 913) would use a comparator similar to that shown in Figure 14 as elements 1415.) coupled to the counter and the plurality of maxtime registers (the "current priority period expired" decision block would require the initialized priority periods values of block 905 and counter be coupled to the comparator in order to carry out the logic), wherein the comparator is configured to compare a count value of the counter with the maxtime value from the plurality of maxtime registers. ("current priority period expired" logic (element 913))

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 12, the rejection of claim 11 has been addressed.
Vaitzblit et al. does not teach a thread selection unit wherein the internal thread counter is incremented and the counter is reset when the count value equals the maxtime value.

Sager teaches a thread selection unit wherein the internal thread counter is incremented (“assign priority to the other thread” (element 917)) and the counter

is reset (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.) when the count value equals the maxtime value. ("the priority period for a thread has expired" [paragraph 51, lines 19-22])

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 13, the rejection of claim 11 has been addressed.

Vaitzblit et al. does not teach a thread selection unit wherein the priority thread counter is incremented and the counter is reset when the maxtime value equals zero.

Sager teaches a thread selection unit wherein the priority thread counter is incremented ("assign priority to the other thread" (element 917)) and the counter is reset (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.) when the maxtime value equals zero. (Figure 9 shows that "At decision loop 913, the method 900 proceeds to block 917 if the current priority duration has expired." [paragraph 51, lines 19-22]. If the counter is zero, the thread would be switched immediately.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of

prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 14, the rejection of claim 12 has been addressed.
Vaitzblit et al. further teaches a thread selection unit wherein the priority thread selector further comprises a priority thread register configured to receive the internal thread value (general purpose ready queue" (element 108) Figure 2,) and the provide the priority thread value (THREAD1 to THREADn (elements 102 to 106) in the general purpose ready queue" (element 108)),

Vaitzblit et al. does not teach a thread selection unit wherein the priority thread registers stores the internal thread value when the maxtime value is not equal to zero.

Sager teaches a thread selection unit wherein the priority thread registers stores the internal thread value when the maxtime value is not equal to zero. (Figure 9 shows that "At decision loop 913, the method 900 proceeds to block 917 if the current priority duration has expired." [paragraph 51, lines 19-22]. If the counter is zero, the thread would be switched immediately.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 19, Vaitzblit et al. teaches a method of selecting an execution thread from a plurality of active threads in a multithreaded processor ([column 3, lines 38-43]), the method comprising:

selecting the priority thread as the execution thread, when the priority thread is unblocked. ([column 5, lines 22-25])

Vaitzblit et al. does not teach selecting a priority thread;

Sager teaches a method of selecting a priority thread; (apparatus for controlling the processing priority [¶61 lines 2-3] and see Figure 13 which can be used to implement the flow diagram of Figure 9)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 20, the rejection of claim 19 has been addressed.

Vaitzblit et al. further teaches a method further comprising selecting a non-priority thread as the execution thread when the priority thread is blocked. ([column 5, lines 22-25])

As per claim 21, the rejection of claim 20 has been addressed.

Vaitzblit et al. does not teach a method wherein the selecting a priority thread comprises: assigning a maxtime value for each active thread; selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles.

Sager teaches a method of selecting a next thread as the priority thread ("assign priority to the other thread" (element 917)) when the priority thread has been the priority thread for a maxtime number of cycles ("current priority period expired" logic (block 913)).

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler).

The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 22, the rejection of claim 21 has been addressed.
Vaitzblit et al. does not teach wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: incrementing a priority thread counter when a count value equals the maxtime value corresponding to the priority thread; and resetting a counter when the count value equals the maxtime value corresponding to the priority thread.

Sager teaches a method wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: of incrementing a priority thread counter (“assign priority to the other thread” (element 917)) when a count value equals the maxtime value (“the priority period for a thread has expired” [paragraph 51, lines

19-22]) corresponding to the priority thread; and resetting a counter when the count value equals the maxtime value (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.) corresponding to the priority thread.

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 23, the rejection of claim 22 has been addressed.

Vaitzblit et al. does not teach wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: incrementing an internal thread counter when a count value equals the maxtime value corresponding to a internal thread value; resetting a counter when the count value equals the maxtime value corresponding to the priority thread; and setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero.

Sager teaches a method wherein the selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: incrementing an internal thread counter ("assign priority to the other thread" (element 917)) when a count value equals the maxtime value corresponding to a internal thread value ("the priority period for a thread has expired" [paragraph 51, lines 19-22]); resetting a counter when the count value equals the maxtime value corresponding to the priority thread (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.); and setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero. (Figure 9 shows that "At decision loop 913, the method 900

proceeds to block 917 if the current priority duration has expired." [paragraph 51, lines 19-22]. If the counter is zero, the thread would be switched immediately.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 24, Vaitzblit et al teaches a thread selection unit for selecting an execution thread (general purpose ready queue" (element 108) Figure 2) from a plurality of active threads ([column 3, lines 38-43]) in a multithreaded processor ([column 3, lines 38-43]), the thread selection unit comprising:

means for selecting the priority thread as the execution thread, when the priority thread is unblocked. ([column 5, lines 22-23])

Vaitzblit et al does not each a means for selecting a priority thread.

Sager teaches a means for selecting a priority thread. (apparatus for controlling the processing priority [¶61 lines 2-3] and see Figure 13 which can be used to implement the flow diagram of Figure 9)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously

change their state in response to the changes in the other threads without doing any useful work.

As per claim 25, the rejection of claim 24 has been addressed.
Vaitzblit et al. further teaches a thread selection unit further comprising means for selecting a non-priority thread as the execution thread when the priority thread is blocked. ([column 5, lines 22-25])

As per claim 26, the rejection of claim 25 has been addressed.
Vaitzblit et al. does not teach wherein the means for selecting a priority thread comprises: means for assigning a maxtime value for each active thread; means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles.

Sager teaches a thread selection unit wherein the means for selecting a priority thread comprises: means for assigning a maxtime value for each active thread; ("initialize priority period for each thread" as found in element 909 in Figure 9) means for selecting a next thread as the priority thread ("assign priority to the other thread" (element 917)) when the priority thread has been the priority thread for a maxtime number of cycles. ("current priority period expired" logic (block 913))

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 27, the rejection of claim 26 has been addressed.
Vaitzblit et al. does not teach wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing a priority thread counter when a count value equals the maxtime value corresponding to the priority thread; and means for resetting a counter when the count value equals the maxtime value corresponding to the priority thread.

Sager teaches a thread selection unit wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing a priority thread counter ("assign priority to the other thread" (element 917)) when a count value equals the maxtime value corresponding to the priority thread. ("the priority period for a thread has expired" [paragraph 51, lines 19-22]) and means for resetting a counter when the count value equals the maxtime value corresponding to the priority thread. (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign

threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously change their state in response to the changes in the other threads without doing any useful work.

As per claim 28, the rejection of claim 26 has been addressed.
Vaitzblit et al. does not teach wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing an internal thread counter when a count value equals the maxtime value corresponding to a internal thread value; means for resetting a counter when the count value equals the maxtime value corresponding to the priority thread; and means for setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero.

Sager teaches a thread selection unit wherein the means for selecting a next thread as the priority thread when the priority thread has been the priority thread for a maxtime number of cycles comprises: means for incrementing an internal thread counter ("assign priority to the other thread" (element 917)) when a count value equals the maxtime value corresponding to a internal thread value; ("the priority period for a thread has expired" [paragraph 51, lines 19-22]). means for resetting a counter when the count value equals the maxtime value

corresponding to the priority thread; (counters are reset in element 1009 of Figure 10 due to a nuke or reset condition [¶54 lines 1-3]. Also, [¶51 lines 1-4] where initializing a priority period is interpreted as resetting.) and means for setting a priority thread value equal to the internal thread value when the maxtime value corresponding to the internal thread value is not equal to zero; (Figure 9 shows that "At decision loop 913, the method 900 proceeds to block 917 if the current priority duration has expired." [paragraph 51, lines 19-22]. If the counter is zero, the thread would be switched immediately.)

It would be obvious to one skilled in the art at the time of the inventions to have the apparatus for controlling the processing priority of Sager (which can implement the flow diagram of Figure 9) output to the Vaitzblit et al. general purpose ready queue (element 108). Note that in the Vaitzblit et al. invention, the general-purpose class [column 3, lines 29-32] of which threads are a part of [column 3, lines 38-43] and Figure 2, element 100 are an input to the scheduler). The prioritization of threads in the general purpose ready queue (element 108) was not a required component to be disclosed in the Vaitzblit et al. invention. The purpose of combining the two inventions would be to provide a means of prioritized thread control to the Vaitzblit et al invention which would assign threads alternately to prevent deadlock and livelock problems between threads often seen in multithreaded systems when two or more threads continuously

change their state in response to the changes in the other threads without doing any useful work.

26. Claims 15-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vaitzblit et al. (U.S. Patent No. 5,528,513, Date Published: June 18, 1996) in view of Sager (U.S. Application No. 10/361,368, Date Published: August 14, 2003) in view of Borkenhagen et al. (U.S. Patent No. 6,567,839, Date Published: May 20, 2003)

As per claim 15, the rejection of claim 1 has been addressed.
Vaitzblit et al. further teaches a thread selection unit wherein the execution thread selector comprises: a thread block checker ([column 5, lines 20-25]) configured to provide a plurality of block values (([column 5, lines 20-25], where it is understood that each task would have its own block value that the scheduler would use to determine if switching tasks is required.)

wherein each active thread has a corresponding block value (([column 5, lines 20-25], where it is understood that each task would have its own block value that the scheduler would use to determine if switching tasks is required.);

Vaitzblit et al. does not teach an execution thread register configured to provide the execution thread value;

Borkenhagen et al. teaches a thread selection an execution thread register (storage control unit (element 200) of Figure 4B, where one skilled in the art would understand that the execution thread register are part of the instruction pipeline for execution of instructions) configured to provide the execution thread value (input "A" in Figures 4A and output "A" in Figure 4B));

It would have been obvious to one skilled in the art at the time of the invention to have used the a similar register design as implementation Borkenhagen et al. to have stored instructions prior to execution by the CPU shown as Vaitzblit et al. element 50 in Figure 1. The purpose would have been to enable the scheduler to continue managing tasks while stored instructions are waiting CPU processing.

Vaitzblit et al. and Borkenhagen et al. do not teach a comparator configured to compare the priority thread value with the execution thread value and to generate a comparison result.
Sager teaches the use of a comparator (comparator similar to that shown in Figure 14 as elements 1415.

It would be obvious to someone skilled in the art at the time of the invention that a comparator similar to that disclosed Sager could be used to compare could be combined with the priority thread value of Vaitzblit et al. and the execution thread

value as disclosed in Borkenhagen et al. to generate a comparison result. The purpose of the comparison would be to more quickly control determination of the next thread to be selected by utilizing the immediate result of the comparison in the logic which the next thread selector outputs.

As per claim 16, the rejection of claim 15 has been addressed.
Vaitzblit et al. further teaches a thread selection unit wherein the execution thread selector further comprises a controller (the scheduler (element 53) is located on a server (element 20) of Figure 1) coupled to receive the block values ([column 5, lines 20-25]), the priority thread value (general purpose ready queue (element 108) shown in Figure 2), and configured to generate a next execution thread (real-time streams [column 3, lines 17-26]) value for the execution thread register.

Vaitzblit et al. does not teach a thread selection unit wherein the execution thread selector further comprises a controller configured to receive the comparison result and the execution thread value
Borkenhagen et al. teaches a thread selection a controller configured to receive the execution thread value. (thread switch controller (element 450) of Figure 4A is shown receiving all inputs required for determining the active thread.)

It would have been obvious to one skilled in the art at the time of the invention to have used the a similar register design as implementation Borkenhagen et al. to have stored instructions prior to execution by the CPU shown as Vaitzblit et al. element 50 in Figure 1. The purpose would have been to enable the scheduler to work more efficiently by continuing to managing tasks while stored instructions are waiting CPU processing.

It would be obvious to one skilled in the art that the controller (scheduler) of Vaitzblit et al. could have access to the execution thread as disclosed in Borkenhagen et al. and to the comparison result as discussed in Claim 15 (for which the office assumes that this claim is dependant). A simple comparator, which would be known to someone skilled in the art, would provide the required comparison result. The purpose of combining all of these inputs into one controller would be to allow the controller access to information required to more efficiently queue the next execution thread, such as switching threads when the active thread is blocked.

As per claim 17, the rejection of claim 16 has been addressed.
Vaitzblit et al. further teaches a thread selection unit wherein the controller generates the next execution thread value to be equal to the priority thread value when the priority thread is not blocked. ([column 5, lines 22-25])

As per claim 18, the rejection of claim 17 has been addressed.

Vaitzblit et al. further teaches a thread selection unit wherein the controller generates the next execution thread value to not be equal to the priority thread value when the priority thread is blocked. ([column 5, lines 22-25])

Conclusion

27. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Borkenhagen et al. (US Patent: 6212544, Date issued: April 3, 2001) – teaches thread switch logic that has a time-out register which forces a thread switch when execution of the active thread in the multithreaded processor exceeds a programmable period of time. Thread switch logic also has a forward progress count register to prevent repetitive unproductive thread switching between threads in the multithreaded processor. Thread switch logic also is responsive to a software manager capable of changing the priority of the different threads and thus superseding thread switch events.
- Borkenhagen et al. (US Patent: 6105051, Date issued: August 15, 2000) – teaches an apparatus and method to guarantee forward progress in execution of threads in a multithreaded processor.
- Flynn et al. (US Patent: 6256775, Date issued: July 3, 2001) – teaches a hardware based thread switch control method, apparatus, and article of manufacture.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Gregory Pollock whose telephone number is 571 270-1465. The examiner can normally be reached on 7:30 AM - 6 PM, Mon-Thu Eastern Time.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Thu Nguyen can be reached on 571 272-6967. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

GAP

12/20/2007

/GAP/

/Gregory Pollock/

Examiner, Art Unit 4182

/Thu Nguyen/
Supervisory Patent Examiner, Art Unit 4182